# IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

**Patent Application**

**Inventors:** Peter Joseph Giacomini et al.

**Serial No.:** 09/725732

**Filing Date:** 11/29/2000

**Art Unit:** 2142

**Examiner:** Thong H Vu

**Docket No.:** 500-001US

**Title:** Distributed Caching Architecture For Computer Networks

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

## SUPPLEMENTAL APPEAL BRIEF UNDER 37 CFR 41.67

Pursuant to 37 CFR 41.67, this brief is filed in support of the appeal in this application.

## TABLE OF CONTENTS

## REAL PARTY IN INTEREST

The real party of interest in this application is the assignee of this application: Broadspider Networks, Inc., formerly of Shrewsbury, New Jersey. The attorneys for this Appeal, DeMont & Breyer, LLC, have an equity interest in the assignee of this application.

## RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences.

## STATUS OF CLAIMS

Claims 1-22 stand rejected and are being appealed.

## STATUS OF AMENDMENTS

All amendments have been entered.

## SUMMARY OF THE CLAIMED SUBJECT MATTER

To make it easier for the average reader to fully understand and appreciate the present invention, the present invention is explained as it might be applied in the context of the World Wide Web. The scope of the claims, however, is not so limited.

Prior Art

When a user of the World Wide Web requests a Web page, the user must wait until the page is available on his or her client (*e.g.*, computer, *etc.*) for viewing. In general, this wait occurs because the request for the Web page and the name of the Web page must traverse the Internet from the client to the Web server that is the source of the page, the request must be fulfilled, and the requested page must travel back to the client. If the Internet is congested or the Web server that is the source of the page is overwhelmed with many concurrent requests for pages, the wait can be considerably long. **(Applicants Specification Page 1, Lines 11-17)**

To shorten this wait, caches are deployed throughout the Internet that store commonly-requested Web pages. The presence of caches in the Internet expedites the delivery of Web pages to clients in two ways. First, a cache eliminates the need for the request to travel all of the way to the Web server, and, therefore, eliminates some the wait associated with the transit. Second, a cache also reduces the number of Web page requests that must be fulfilled by the Web server, and, therefore, reduces the wait associated with contention for the Web server. **(Applicants Specification Page 1, Lines 18-28)**

The information that a cache stores, and the mechanism that it uses to index that information is unintuitive. In order to understand this, let's talk about how a typical paper telephone book works.

A "normal" telephone book provides a mapping of an alphabetized list of names to telephone numbers and is said to be indexed by name, as shown in Table 1.

| Name | Telephone Number |
|---|---|
| Orange, William | 545-223-4342 |
| Rehnquist, William | 103-554-2232 |
| Schweitzer, Albert | 222-555-1212 |
| Taylor, Zachary | 212-443-4523 |
| Tell, William | 344-244-4444 |

**Table 1 — Telephone Directory Indexed by Name**

In other words, given a name, you can quickly and easily find the telephone number associated with the name. In contrast, given a telephone number and a telephone book that is indexed by name, you could find the name associated with telephone number by examining the telephone numbers, but the process is tediously slow and might require that you examine all of the telephone numbers in the book. **(Applicants Specification Page 1, Lines 18-28)**

A "reverse" telephone book is a mapping of an ordinal list of telephone numbers to names and is said to be indexed by telephone number, as shown in Table 2.

| Telephone Number | Name |
|---|---|
| 103-554-2232 | Rehnquist, William |
| 212-443-4523 | Taylor, Zachary |
| 222-555-1212 | Schweitzer, Albert |
| 344-244-4444 | Tell, William |
| 545-223-4342 | Orange, William |

**Table 2 — Telephone Directory Indexed by Telephone Number**

In other words, given a telephone number, you can quickly and easily find the name associated with the telephone number. In contrast, given a name and a telephone book indexed by number, you could find the telephone number associated with name by examining the names, but the process is tediously slow and might require that you examine all of the names in the book. **(Applicants Specification Page 1, Lines 18-28)**

Here comes the unintuitive part. You might think that a cache of Web pages is indexed by name like a normal telephone book, but it is not. Why? Because the names of Web pages have different lengths, as measured in bits, and when the names have different widths, their storage in a computer is inefficient. A typical cache almost always runs nears its capacity, and, therefore, anything that can be done to improve this efficiency is advantageous.

To prevent this inefficiency, a cache of Web pages is indexed by a string that is a function or "nickname" of the name of the Web page. The telephone directory in Table 1 indexed by nickname appears in Table 3.

| Nickname | Telephone Number |
|---|---|
| Al | 222-555-1212 |
| Bill | 545-223-4342 |
| Bill | 103-554-2232 |
| Bill | 344-244-4444 |
| Zeke | 212-443-4523 |

**Table 3 — Telephone Directory Indexed by Nickname**

A problem with nicknames is that many different people often have the same nickname and it isn't possible, using nicknames alone, to distinguish William Rehnquist's telephone number from William of Orange's or William Tell's (See Table 3).

To ensure that the cache of Web pages does not suffer the same problem, *the cache stores both the Web page and the name of the Web page together and the combination is indexed by the nickname string*. This enables the cache to distinguish or "disambiguate" between Web pages whose names have the same nickname.

A telephone directory like this appears in Table 4.

| Nickname | Name | Telephone Number |
|---|---|---|
| Al | Schweitzer, Albert | 222-555-1212 |
| Bill | Orange, William | 545-223-4342 |
| Bill | Rehnquist, William | 103-554-2232 |
| Bill | Tell, William | 344-244-4444 |
| Zeke | Taylor, Zachary | 212-443-4523 |

**Table 4 — Telephone Directory Indexed by Nickname and Disambiguated by Name**

William Rehnquist's telephone number can be obtained from the telephone directory in Table 4 first by computing his nickname "Bill" and then by using his full name to disambiguate among the three entries for Bill.

Although caches for Web pages use this mechanism, the process of disambiguation takes time. Therefore, a function that reduces the likelihood that the same nickname string will be generated from two different Web page names is preferred. Such functions have unusual mathematical properties and are known as "hash functions." The "nickname" strings that hash functions generate are called "hash keys."

Now it is time to see how all of this works together. In the prior art, when a user of a client machine desires a Web page, the name or "URL" of the Web page traverses the Internet from the client machine to the cache of Web pages. (It is not germane to the present invention what pages are stored in the cache or how they got there.) The cache uses the hash function on the name of the Web page to generate the hash key or nickname of the Web page. The cache then uses the hash key as the index into its database to find the Web page. If there is more than one entry in the database with the same name the cache uses the name of the Web page to disambiguate among the entries. The cache then sends the proper Web page back to the user's client for viewing.

The Invention In General

The inventors of the present invention recognized that caches in the prior art were spending too many computational resources performing the hash and disambiguation functions and that those functions could be more efficiently handled by the user's client machine. **(Applicants Specification Page 3, Lines 25-31)** In all cases, this relieves the cache of the burden of computing the hash key, and, in some cases, also relieves the cache of the disambiguation function.

Discussion of Each Independent Claim

There are five distinct variations of the present invention that are claimed and taught in the disclosure. Each of the five variations will be described — with reference to the specification by page and line number and to the drawings — and related to the claims. The following table relates the variations, portions of the specification, figures, and claims. In summary:

| Invention Variation | Specification | Principal Supporting Figure | Independent Method Claim | Independent Apparatus Claim |
|---|---|---|---|---|
| #1 | Page 9, Line 1 to Page 11, Line 29 | Fig. 6 | 1, 15 | 8, 16 |
| #2 | Page 11, Line 30 to Page 13, Line 27 | Fig. 7 | 1, 15 | 8, 16 |
| #3 | Page 13, Line 28 to Page 16, Line 21 | Fig. 8, 13 | 21 | 22 |
| #4 | Page 16, Lines 22 to Page 18, Line 19 | Fig. 9, 13 | 21 | 22 |
| #5 | Page 18, Line 20 to Page 32, Line 30 | Fig. 10, 11, 12 | 19 | 20 |

Invention Variation #1 — Claims 1, 8, 15, and 16 (Figure 6)

In accordance with the first variation, a *Client Node* requests a resource from its *Cache Node* and it is the *Client Node* that performs the hashing (rather than the *Cache Node* as in the prior art). In the first variation, the *Cache Node* performs the disambiguation function based on a comparison of the resource identifiers. **(Applicants Specification Page 9, Line 1 to Page 11, line 29; Figures 4, 5, and 6)** Claims 1 and 8 read on the *Client Node*, and claims 15 and 16 read on the *Cache Node*.

Invention Variation #2 — Claims 1, 8, 15, and 16 (Figure 7)

The second variation differs from the first variation in that it is the *Client Node* and not the *Cache Node* that:

i. resolves hash collisions,
ii. determines if the *Cache Node* has provided the requested resource, and
iii. initiates the retrieval of the requested resource from its *the Cache Node's Parent Node* if the *Cache Node* has not provided the requested resource.

The second variation is advantageous over the first variation in that the computational task of resolving hash collisions is moved from the *Cache Node*, which might be computationally taxed, to the *Client Node*, which is more likely than the *Cache Node* to have spare computational capacity. **(Applicants Specification Page 11, Lines 30 to Page 13, Line 27; Figures 4, 5, and 7)** Claims 1 and 8 read on the *Client Node*, and claims 15 and 16 read on the *Cache Node*.

Invention Variation #3 — Claims 21 and 22 (Figure 8)

The third variation of the operation of the illustrative embodiment, in which a given node, hereinafter called the "*Client Node*," requests a resource from its parental node, hereinafter called the "*Cache Node*." The third variation is like the first variation in that it is the *Cache Node* that:

i. resolves hash collisions, and
ii. unilaterally initiates the retrieval of the requested resource from *its* parental node, hereinafter called the "*the Cache Node's Parent Node*," if the *Cache Node* does not have it.

The salient difference between the third variation and the first variation is that the third variation uses two different hash keys, both of which are based on the resource identifier.

The second hash key is used as the index into the cache data structure, as in the first and second variations, and the first hash key is used to resolve hash collisions, whereas in the first and second variations the resource identifier itself is used to resolve hash collisions. An advantage of the third variation over the first variation is that the character length of the first hash key is known and fixed, and, therefore, the computational complexity of the hash collision resolution comparison is known and fixed. A disadvantage of the third variation over the first variation is that it requires the *Client Node* to perform two distinct hash functions (or one larger one whose output is bifurcated into two keys) rather than just one. **(Applicants Specification Page 13, Lines 28 to Page 16, Line 21; Figures 4, 5, 8, and 13)** Claims 21 and 22 read on the *Cache Node*.

Invention Variation #4 — Claims 21 and 22 (Figure 9)

The fourth variation of the operation of the illustrative embodiment, in which a given node, hereinafter called the "*Client Node*," requests a resource from its parental node, hereinafter called the "*Cache Node*." The fourth variation differs from the third variation in that it is the *Client Node* that:

i. resolves hash collisions,
ii. determines if the *Cache Node* has provided the requested resource, and
iii. initiates the retrieval of the requested resource from its *the Cache Node's Parent Node* if the *Cache Node* has not provided the requested resource.

The fourth variation is advantageous over the third variation because the computational task of resolving hash collisions is moved from the *Cache Node*, which might be computationally taxed, to the *Client Node*, which is more likely than the *Cache Node* to have spare computational capacity. **(Applicants Specification Page 16, Lines 22 to Page 18, Line 19; Figures 4, 5, 8, and 13)** Claims 21 and 22 read on the *Cache Node*.

Invention Variation #5 — Claims 19 and 20 (Figure 10)

The fifth variation of the operation of the illustrative embodiment, in which a given node, hereinafter called the "*Client Node*," requests a resource from its parental node, hereinafter called the "*Cache Node*."

The fifth variation differs from the first four variations in that the *Client Node* comprises a "Table of Cached Resources," which is a table that indicates which hash keys that the *Client Node* might generate correspond to resources that have been already cached in the *Cache Node*. In other words, the *Client Node* is capable of conclusively determining if

the *Cache Node's* cache does not contain a resource corresponding to a given hash key, and, therefore should retrieve the resource directly. It should be understood, however, that the converse is not true. The Table of Cached Resources merely indicates that a resource corresponding to a given hash key is cached, but not that it is *the* resource corresponding to the resource identifier. This asymmetry results because different resource identifiers that represent different resources can hash to the same hash key.

In accordance with the fifth variation, it is the *Client Node* that:

i. resolves hash collisions,

ii. determines if the *Cache Node* has provided the requested resource, and

iii. initiates the retrieval of the requested resource from its *the Cache Node's Parent Node* if the *Cache Node* has not provided the requested resource.

Furthermore, in the fifth variation, it is the *Client Node* that bears the ultimate responsibility for ensuring that a resource, however and from whomever retrieved, is the resource corresponding to the resource identifier. The fifth variation is advantageous because many of the computational tasks are performed by the *Client Node*, which is more likely than the *Cache Node* to have spare computational capacity. **(Applicants Specification Page 18, Line 20 to Page 32, Line 30; Figures 4, 5, 10, 11, and 12)** Claims 19 and 20 read on the *Client Node*.

## GROUNDS OF APPEAL OF THE CLAIMED SUBJECT MATTER

**Ground 1: 35 U.S.C 101 Rejection of Claim 15**

Claim 15 was rejected under 35 U.S.C. 101 because "the claimed invention lacks patentable utility.

**Ground 2: 35 U.S.C. 112 Rejection of Claim 16**

Claim 16 was rejected under 35 U.S.C. 112, Second Paragraph, for failing to particularly point out and distinctly claim the subject matter which the applicant regards as the invention.

**Ground 3: 35 U.S.C. 102 Rejection of Claims 1-20**

Claims 1-20 were rejected under 35 U.S.C. 102(b) as being anticipated by P.R. Carpentier, U.S. Patent 6,807,632 (hereinafter "Carpentier").

**Ground 4: 35 U.S.C. 103 Rejection of Claims 21 and 22**

Claims 21 & 22 were rejected under 35 U.S.C. 103(a) as being unpatentable over P.R. Carpentier, U.S. Patent 6,807,632 (hereinafter "Carpentier") in view of J. Takahashi, U.S. Patent 5,428,774 (hereinafter "Takahashi").

## ARGUMENTS

**Ground 1: 35 U.S.C. 101 Rejection of Claim 15**

Claim 15 has been rejected under 35 U.S.C. 101 because the claimed invention lacks "patentable utility." The applicants respectfully traverse.

Claim 15 recites:

> **15.** (Original) A method comprising:
>
> *receiving* a request for a first resource and a hash key that is a hashed function of a first resource identifier;
>
> *retrieving* said first resource and said first resource identifier from a data structure that is indexed by said hash key; and
>
> *transmitting* said first resource and said first resource identifier in response to said request for said first resource.
>
> (*emphasis supplied*)

Claim 15 recites the receiving, retrieving, and transmission of various items, and, as such, achieves a "useful, concrete, and tangible result" in accordance with the Patent Office's Interim Guidelines for Examiner of Patent Applications for Patent Subject Matter Eligibility. This is true whether the request, hash key, resource identifiers, and resources are, for example, electromagnetic signals, papers, or papyrus scrolls. For this reason, the applicants respectfully submit that claim 15 recites patentable subject matter within the requirement of 35 USC 101 and that the rejection is traversed.

**Ground 2: 35 U.S.C. 112 Rejection of Claim 16**

Independent claim 16 was rejected under 35 U.S.C. 112, Second Paragraph, for failing to particularly point out and distinctly claim the subject matter which the applicant regards as the invention. The Office action states something about the claim reciting a "first processor" and a "second processor" and that being confusing, but it doesn't, and, therefore, the applicants respectfully traverse the rejection.

Claim 16 recites:

> **16.** (Original) An apparatus comprising:
>
> a receiver for receiving a request for a first resource and a hash key that is a hashed function of a first resource identifier;
>
> a processor for retrieving said first resource and said first resource identifier from a data structure that is indexed by said hash key; and
>
> a transmitter for transmitting said first resource and said first resource identifier in response to said request for said first resource.
>
> (*emphasis supplied*)

There is no vagueness or ambiguity of the scope of claim 16, and, therefore, the applicants respectfully submit that the rejection of claim 16 is traversed.

**Ground 3: 35 U.S.C. 102 Rejection of Claims 1-20**

Claims 1-20 were rejected under 35 U.S.C. 102(b) as being anticipated by P.R. Carpentier, U.S. Patent 6,807,632 (hereinafter "Carpentier"). The applicants respectfully traverse.

Claim 1 recites:

> **1.** (Original) A method comprising:
>
> *hashing at a first processor a first resource identifier to create a hash key, wherein said first resource identifier identifies a first resource*;
>
> transmitting from said first processor to a second processor said hash key and a request for said first resource; and
>
> receiving at said first processor a second resource in response to the transmission of said hash key and said request for said first resource.
>
> (*emphasis added*)

Nowhere does Carpentier teach or suggest, alone or in combination with the other references, what claim 1 recites — namely, hashing the "name" of the resource to create a "nickname" of the resource.

Carpentier teaches that the more or less arbitrary naming of files that we all use today or what he calls "potentially misleading extrinsic naming" leads to several problems on computer systems. These include:

(1) How do you know when two files with the same name are identical or not?

(2) How do you know when two files with different names are identical or not?

(3) How do you prevent duplicate copies of the same file from taking resources when it is not necessary?

Carpentier's answer is to replace the "normal" name that files are typically given with a digital signature or fingerprint for the file, which is based on the contents of that file. Done properly, this virtually ensures that:

(1) Two files with identical fingerprints are identical, and

(2) Two files with different fingerprints are different.

As those skilled in the art would know, the answer is to use a hash function of the file to generate the fingerprint.

The result is that when Carpentier wants a file from a remote system, which indexes the files based on their digital fingerprint, Carpentier transmits the digital fingerprint of the file to the remote system alone with a request for the file. As advantageous as the teaching of Carpentier might be, nowhere does it teach or suggest, alone or in combination with the other references, what is recited in claim 1. For these reasons, the applicants respectfully submit that the rejection of claim 1 is traversed.

Because claims 2-7 depend on claim 1, the applicants respectfully submit that the rejection of them is also traversed.

Independent claim 8 recites:

> **8.** (Original) An apparatus comprising:
>
> a first processor for *hashing a first resource identifier to create a hash key, wherein said first resource identifier identifies a first resource*;
>
> a transmitter for transmitting said hash key and a request for said first resource to a second processor; and
>
> a receiver for receiving a second resource in response to the transmission of said hash key and said request for said first resource.
>
> (*emphasis supplied*)

Nowhere does Carpentier teach or suggest, alone or in combination with the other references, what claim 8 recites — namely, hashing the "name" of the resource to create a "nickname" of the resource. For these reasons, the applicants respectfully submit that the rejection of claim 8 is traversed.

Because claims 9-14 depend on claim 8, the applicants respectfully submit that the rejection of them is also traversed.

Independent claim 15 recites:

> **15.** (Original) A method comprising:
>
> receiving a request for a first resource and *a hash key that is a hashed function of a first resource identifier*;
>
> retrieving said first resource and said first resource identifier from a data structure that is indexed by said hash key; and
>
> transmitting said first resource and said first resource identifier in response to said request for said first resource.
>
> (*emphasis supplied*)

Nowhere does Carpentier teach or suggest, alone or in combination with the other references, what claim 15 recites — namely, hashing the "name" of the resource to create a "nickname" of the resource. For these reasons, the applicants respectfully submit that the rejection of claim 15 is traversed.

Independent claim 16 recites:

> **16.** (Original) An apparatus comprising:
>
> a receiver for receiving a request for a first resource and *a hash key that is a hashed function of a first resource identifier;*
>
> a processor for retrieving said first resource and said first resource identifier from a data structure that is indexed by said hash key; and
>
> a transmitter for transmitting said first resource and said first resource identifier in response to said request for said first resource.
>
> (*emphasis supplied*)

Nowhere does Carpentier teach or suggest, alone or in combination with the other references, what claim 16 recites — namely, hashing the "name" of the resource to create a "nickname" of the resource. For these reasons, the applicants respectfully submit that the rejection of claim 16 is traversed.

Independent claim 17 recites:

> **17.** (Previously Presented) A method comprising:
>
> receiving from a first processor a first resource identifier that identifies a first resource, *a hash key that is a hashed function of said first resource identifier*, and a request for a first resource;
>
> retrieving a second resource and a second resource identifier from a data structure that is indexed by said hash key;
>
> verifying that said second resource is said first resource by comparing said second resource identifier to said first resource identifier; and
>
> transmitting said second resource to said first processor when said second resource is verified as said first resource.

(*emphasis supplied*)

Nowhere does Carpentier teach or suggest, alone or in combination with the other references, what claim 17 recites — namely, hashing the "name" of the resource to create a "nickname" of the resource. For these reasons, the applicants respectfully submit that the rejection of claim 17 is traversed.

Independent claim 18 recites:

> **18.** (Previously Presented) An apparatus comprising:
>
> a receiver for receiving from a first processor a first resource identifier that identifies a first resource, *a hash key that is a hashed function of said first resource identifier*, and a request for a first resource;
>
> a second processor for retrieving a second resource and a second resource identifier from a data structure that is indexed by said hash key, and for verifying that said second resource is said first resource by comparing said second resource identifier to said first resource identifier; and
>
> a transmitter for transmitting said second resource to said first processor when said second resource is verified as said first resource.
>
> (*emphasis supplied*)

Nowhere does Carpentier teach or suggest, alone or in combination with the other references, what claim 18 recites — namely, hashing the "name" of the resource to create a "nickname" of the resource. For these reasons, the applicants respectfully submit that the rejection of claim 18 is traversed.

Independent claim 19 recites:

> **19.** (Original) A method comprising:
>
> hashing at a first processor *a first resource identifier to create a hash key, wherein said first resource identifier identifies a first resource*;
>
> transmitting from said first processor to a second processor said hash key and a request for said first resource when said all or a portion of said hash key is contained in a list of valid hash keys associated with said first processor; and
>
> receiving at said first processor said first resource in response to the transmission of said hash key and said request for said first resource.
>
> (*emphasis supplied*)

Nowhere does Carpentier teach or suggest, alone or in combination with the other references, what claim 19 recites — namely, hashing the "name" of the resource to create a "nickname" of the resource. For these reasons, the applicants respectfully submit that the rejection of claim 19 is traversed.

Independent claim 20 recites:

> **20.** (Previously Presented) An apparatus comprising:
>
> a first processor for *hashing a first resource identifier to create a hash key*, wherein said first resource identifier identifies a first resource, and for verifying that all or a portion of said hash key is contained in a list of valid hash keys;
>
> a transmitter for transmitting from said first processor to a second processor said hash key and a request for said first resource; and
>
> a receiver for receiving said first resource in response to the transmission of said hash key and said request for said first resource.
>
> (*emphasis supplied*)

Nowhere does Carpentier teach or suggest, alone or in combination with the other references, what claim 20 recites — namely, hashing the "name" of the resource to create a "nickname" of the resource. For these reasons, the applicants respectfully submit that the rejection of claim 20 is traversed.

**Ground 4: 35 U.S.C. 103 Rejection of Claims 21 and 22**

Claims 21 & 22 were rejected under 35 U.S.C. 103(a) as being unpatentable over P.R. Carpentier, U.S. Patent 6,807,632 (hereinafter "Carpentier") in view of J. Takahashi, U.S. Patent 5,428,774 (hereinafter "Takahashi"). The applicants traverse.

Independent claim 21 recites:

> **21.** (Previously Presented) A method comprising:
>
> receiving from a first processor a request for a first resource and *a first hash key that is a hashed function of a first resource identifier*;
>
> retrieving a second resource and a first portion of a second hash key from a data structure that is indexed by a first portion of said first hash key;
>
> verifying that said second resource is said first resource by comparing a second portion of said first hash key to said first portion of said second hash key; and
>
> transmitting said second resource to said first processor when said second resource is verified as said first resource.
>
> (*emphasis supplied*)

Nowhere does Carpentier or Takahashi teach or suggest, alone or in combination with the other references, what claim 21 recites — namely, hashing the "name" of the resource to create a "nickname" of the resource. For these reasons, the applicants respectfully submit that the rejection of claim 21 is traversed.

Independent claim 22 recites:

> **22.** (Previously Presented) An apparatus comprising:
>
> a receiver for receiving from a first processor a request for a first resource and a first hash key that is a hashed function of a first resource identifier;
>
> a second processor for retrieving a second resource and a first portion of a second hash key from a data structure that is indexed by a first portion of said first hash key, and for verifying that said second resource is said first resource by comparing a second portion of said first hash key to said first portion of said second hash key; and
>
> a transmitter for transmitting said second resource to said first processor when said second resource is verified as said first resource.
>
> (*emphasis supplied*)

Nowhere does Carpentier or Takahashi teach or suggest, alone or in combination with the other references, what claim 22 recites — namely, hashing the "name" of the resource to create a "nickname" of the resource. For these reasons, the applicants respectfully submit that the rejection of claim 22 is traversed.

## CONCLUSION

The applicants have demonstrated that the logic underlying the Office's rejection is untenable, and, therefore, that the rejection is not sustainable. For this reason, the applicants respectfully request the Board of Appeals to reverse the decision of the Examiner as provided for in 37 C.F.R. 41.50(a).

Respectfully,
Peter Joseph Giacomini et al.

By **/Jason Paul DeMont/**
Jason Paul DeMont
Reg. No. 35793
Attorney for Applicants
732-578-0103 x11

DeMont & Breyer, L.L.C.
Suite 250
100 Commons Way
Holmdel, NJ 07733
United States of America

## CLAIMS APPENDIX

**1.** (Original) A method comprising:

hashing at a first processor a first resource identifier to create a hash key, wherein said first resource identifier identifies a first resource;

transmitting from said first processor to a second processor said hash key and a request for said first resource; and

receiving at said first processor a second resource in response to the transmission of said hash key and said request for said first resource.

**2.** (Original) The method of claim 1 further comprising receiving at said first processor a second resource identifier in response to the transmission of said hash key and said request for said first resource.

**3.** (Original) The method of claim 2 wherein said first processor verifies that said second resource is said first resource by comparing said second resource identifier to said first resource identifier.

**4.** (Original) The method of claim 1 further comprising transmitting from said first processor to said second processor said first resource identifier in addition to said hash key and said request for said first resource.

**5.** (Original) The method of claim 4 wherein said second processor stores said second resource and said second resource identifier in a data structure that is indexed by said hash key.

**6.** (Original) The method of claim 5 wherein said second processor verifies that said second resource is said first resource by comparing said second resource identifier to said first resource identifier.

**7.** (Original) The method of claim 1 wherein said hash key and said request for said first resource are transmitted from said first processor to said second processor when said all or a portion of said hash key is contained in a list of valid hash keys associated with said first processor.

**8.** (Original) An apparatus comprising:

a first processor for hashing a first resource identifier to create a hash key, wherein said first resource identifier identifies a first resource;

a transmitter for transmitting said hash key and a request for said first resource to a second processor; and

a receiver for receiving a second resource in response to the transmission of said hash key and said request for said first resource.

**9.** (Original) The apparatus of claim 8 wherein said receiver also receives a second resource identifier in response to the transmission of said hash key and said request for said first resource.

**10.** (Original) The apparatus of claim 9 wherein said first processor verifies that said second resource is said first resource by comparing said second resource identifier to said first resource identifier.

**11.** (Original) The apparatus of claim 8 wherein said transmitter also transmits to said second processor said first resource identifier in addition to said hash key and said request for said first resource.

**12.** (Original) The apparatus of claim 11 wherein said second processor stores said second resource and said second resource identifier in a data structure that is indexed by said hash key.

**13.** (Original) The apparatus of claim 12 wherein said second processor verifies that said second resource is said first resource by comparing said second resource identifier to said first resource identifier.

**14.** (Original) The apparatus of claim 8 wherein said hash key and said request for said first resource are transmitted from said first processor to said second processor when said all or a portion of said hash key is contained in a list of valid hash keys associated with said first processor.

**15.** (Original) A method comprising:

receiving a request for a first resource and a hash key that is a hashed function of a first resource identifier;

retrieving said first resource and said first resource identifier from a data structure that is indexed by said hash key; and

transmitting said first resource and said first resource identifier in response to said request for said first resource.

**16.** (Original) An apparatus comprising:

a receiver for receiving a request for a first resource and a hash key that is a hashed function of a first resource identifier;

a processor for retrieving said first resource and said first resource identifier from a data structure that is indexed by said hash key; and

a transmitter for transmitting said first resource and said first resource identifier in response to said request for said first resource.

**17.** (Previously Presented) A method comprising:

receiving from a first processor a first resource identifier that identifies a first resource, a hash key that is a hashed function of said first resource identifier, and a request for a first resource;

retrieving a second resource and a second resource identifier from a data structure that is indexed by said hash key;

verifying that said second resource is said first resource by comparing said second resource identifier to said first resource identifier; and

transmitting said second resource to said first processor when said second resource is verified as said first resource.

**18.** (c) An apparatus comprising:

a receiver for receiving from a first processor a first resource identifier that identifies a first resource, a hash key that is a hashed function of said first resource identifier, and a request for a first resource;

a second processor for retrieving a second resource and a second resource identifier from a data structure that is indexed by said hash key, and for verifying that said second resource is said first resource by comparing said second resource identifier to said first resource identifier; and

a transmitter for transmitting said second resource to said first processor when said second resource is verified as said first resource.

**19.** (Original) A method comprising:

hashing at a first processor a first resource identifier to create a hash key, wherein said first resource identifier identifies a first resource;

transmitting from said first processor to a second processor said hash key and a request for said first resource when said all or a portion of said hash key is contained in a list of valid hash keys associated with said first processor; and

receiving at said first processor said first resource in response to the transmission of said hash key and said request for said first resource.

**20.** (Previously Presented) An apparatus comprising:

a first processor for hashing a first resource identifier to create a hash key, wherein said first resource identifier identifies a first resource, and for verifying that all or a portion of said hash key is contained in a list of valid hash keys;

a transmitter for transmitting from said first processor to a second processor said hash key and a request for said first resource; and

a receiver for receiving said first resource in response to the transmission of said hash key and said request for said first resource.

**21.** (Previously Presented) A method comprising:

receiving from a first processor a request for a first resource and a first hash key that is a hashed function of a first resource identifier;

retrieving a second resource and a first portion of a second hash key from a data structure that is indexed by a first portion of said first hash key;

verifying that said second resource is said first resource by comparing a second portion of said first hash key to said first portion of said second hash key; and

transmitting said second resource to said first processor when said second resource is verified as said first resource.

---

**22.** (Previously Presented) An apparatus comprising:

a receiver for receiving from a first processor a request for a first resource and a first hash key that is a hashed function of a first resource identifier;

a second processor for retrieving a second resource and a first portion of a second hash key from a data structure that is indexed by a first portion of said first hash key, and for verifying that said second resource is said first resource by comparing a second portion of said first hash key to said first portion of said second hash key; and

a transmitter for transmitting said second resource to said first processor when said second resource is verified as said first resource.

## EVIDENCE APPENDIX

There is no evidence submitted pursuant to 37 CFR §§ 1.130, 1.131, or 1.132.

## RELATED PROCEEDINGS INDEX

There are no related proceedings.